IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named Applicant: Lotspiech	)	Art Unit: 2136
Serial No.: 09/609,809	)	Examiner: Colin
Filed: July 3, 2000	)	**ARC9-2000-0063-US1**
For: **FAULT INTOLERANT CIPHER CHAINING**	)	July 27, 2004 750 B STREET, Suite 3120 San Diego, CA 92101

APPEAL BRIEF

This appeal brief is submitted under 35 U.S.C. §134. This appeal is further to Appellant's Notice of Appeal filed herewith.

Table of Contents

(1) Real Party in Interest

The real party in interest is IBM Corp.

(2) Related Appeals/Interferences

No other appeals or interferences exist which relate to the present application or appeal.

(3) Status of Claims

Claims 6-17 are pending and finally rejected.

(4) Status of Amendments

An amendment cancelling Claims 1-5 has been submitted and presumably will be entered for purposes of appeal.

(5) Summary of Invention

Using Claim 6 as an example, the invention is a computer program device that includes logic means for chaining a data block to a plain text version of an adjacent block in the stream to render a chained block, and logic means for scrambling the chained block using a first round of a cipher to render a scrambled block, page 3, first full paragraph and Figure 2. Logic means are provided for iterating the means for scrambling and chaining using subsequent rounds of the cipher, id..

On the other hand, Claim 8 recites a computer system for encrypting a stream of data blocks. The system includes a processor programmed to execute method acts, page 3, second paragraph and Figure 2. The method acts executed by the processor include receiving a sequence of N blocks, and then, for i= 1 to N, executing a DO loop in a forward chaining process.

The forward chaining DO loop includes an XOR step, where an ith block is XORed with the result of the XOR step on block i-1, pages 7-8, Figure 2. The XOR step is followed by a scrambling step, in which one round of a cipher is performed to scramble the result of the XOR, pages 7-8, Figure 2. The scrambling step is followed by a determination of whether block i+1 exists. If it does, the output of the XOR step of block i is XORed with block i+1, i is incremented by unity, and the chaining process continues, pages 7-9, Figure 2.

In contrast, when it is determined that a $block_{i+1}$ does not exist, the method acts executed by the processor executing a DO loop for i = N to 1 in a backward chaining process, id. The backward chaining process includes an XOR step, where an ith block is XORed with the result of the XOR step from block i+1. The XOR step is followed by a scrambling step, in which one round of a cipher is performed to scramble the result of the XOR. The scrambling step is followed by a determination whether block i-1 exists. If it does, the output of the XOR step of block i is XORed with block i-1, i is decremented by unity, and the chaining process continues. Otherwise, it is determined whether a predetermined number of iterations have been executed, and if not, another forward chaining loop is executed using a next round of the cipher. When all cipher rounds have been used, an encrypted stream of data blocks is output.

(6) Issues

(a) Whether Claims 6, 7, 11, and 13 are unpatentable under 35 U.S.C. §102 as being anticipated by Zhang, USPN 6,154,541.

(b) Whether Claims 8-10, 12, 14, and 15-17 are unpatentable under 35 U.S.C. §103 as being obvious over Zhang.

(7) Grouping of Claims

The rejected claims are grouped as indicated above owing to the different grounds of rejection applied to each.

(8a) Argument

The rejection reflects an erroneous factual reading of Zhang and thus constitutes reversible error. Specifically, Zhang seeks to solve the so-called "code book" attack, in which even if the key or cipher is

unknown, if the same words always produce the same ciphertext a frequency analysis can be performed to deduce what the words mean. This attack is defeated by Zhang's cipher block chaining, in which a word's scrambled version is always different because it depends on what preceded it in the message. Zhang, because it uses a pseudo one-time pad, further embellishes the code book attack solution by performing forward and backward chaining, so that even the first part of the message cannot be defeated by the code book attack. Importantly, the relied-upon portion of Zhang does one thing - chaining, using cipher chaining instead of plain text chaining - and not two things, namely, both chaining and, within the chaining process, running rounds of a cipher.

In contrast, the claims rejected under this section not only require both chaining and running rounds of a cipher, they recite alternating the running of the rounds of the cipher with running the chaining mode, something never before done or suggested to the best of Applicant's knowledge and certainly not suggested in the references of record. For instance, Claim 13 requires scrambling a block using one and only one round of a cipher, ***then*** chaining the block to another block to render a chained block, ***then*** scrambling the chained block using one and only round of the cipher, something that the relied-upon section of Zhang does not do. Instead, the relied-upon section of Zhang performs all of the chaining without scrambling between chainings. The fact that Zhang uses cipher chaining and not plain text chaining should not confuse the reader that Zhang scrambles within rounds of chainings, when in fact it does not do so.

The examiner has refused to accept the above explanation of Zhang, offered in consultation with the present inventor, a well-known expert in the field. Instead of recognizing right answer when told he has responded to the above points by demonstrating the confusion alluded to above. Specifically, on page 3 of the Office Action dated April 27, 2004, the examiner alleges in a somewhat garbled argument that Zhang,

col. 23, line 30 through col. 24, line 52 teaches backward and forward scrambling and backward and forward chaining. On the contrary, the first method taught in this section (backward and forward scrambling, or BFSM) at col. 23, line 35 through col. 24, line 5 simply discloses backward and forward scrambling on already-encrypted data, col. 23, line 36. One of the scrambling functions may be LZW compression; DES may also be used.

The second method (expanded data block method or EDBM) is briefly mentioned at col. 24, lines 6-17. Neither method in these sections of Zhang mentions chaining a data block to a plain text version of an adjacent block in the stream to render a chained block, much less that it is the chained block that is scrambled, much less still that the scrambling and chaining are iterated.

Beginning at col. 24, line 22 Zhang mentions that "feedback mode (stream cipher feedback or block chaining), which is some type of scrambling in essence, can also be used at various levels" [referring not to various levels of the BFSM process but to granularity of data that can be chained]. Zhang then goes on to describe what he means, but does not describe just how the "feedback mode" is to be integrated with BFSM until line 47 of col. 24, which indeed is telling for what it appears to teach:

> "An alternate form of applying BFSM is to make use of feedback mode. Since feedback is in essence a type of scrambling in a chaining fashion, it can be utilized ***as one of the bi-directional scrambling operations***, e.g., data are first scrambled backward with one type of scrambling and the scrambled version ***is then encrypted*** in the feedback mode." (emphasis mine).

This proves Appellant's point. To the extent that Zhang contemplates block chaining, it is only as a ***complete replacement*** for one of its scrambling operations, not as an operation that is integrated within iterations of a chain-scramble-chain-scramble scheme as recited in, e.g., Claims 6, 11, and 13. Simply put,

the rejection of the present claims based on Zhang cannot be sustained, because as discussed above, just because Zhang mentions both of the terms "chaining" and "scrambling" does not mean that it teaches chaining and scrambling applied together in the manner particularly claimed, which indeed it fails to do.

(8b)

For the reasons stated above, the claims rejected for being obvious over Zhang are patentable. Zhang does not come close to Claim 8, for instance. The Board will not fail to notice that in rejecting Claim 8, the examiner alleges that several things are present in Zhang that in fact are not there. As an example, page 8 of the Office Action alleges that the initialized block variable "B" is taught in Zhang, col. 23, lines 48-50, and that B is XOR'ed with in i^{th} block and then set equal to the i^{th} block at col. 23, lines 47-50. But in reality the relied-upon section of Zhang simply teaches that an initial "key" is used as the first encryption to input into a DES function wherein previous bits are used ***as a key to encrypt*** following bits. No "XOR" is mentioned, much less that "B" is ever set equal to an i^{th} block. Likewise, the allegations that Zhang teaches the portions of Claim 8 that are directed to iteratively alternating between chaining (by XORing) and scrambling appear to be based on fantasy. They certainly are not present in Zhang where the examiner says they are. More is required to reject a claim than an imagination made vivid by Appellant's own specification.

Furthermore, in a prior rejection Zhang had been combined with Enichen, but when Enichen had been disqualified as prior art, the examiner simply alleged that the now-missing pieces from Zhang are obvious, without any evidence of record to support this contention. Absent evidence to support a finding of fact, the rejection is reversible error.

Respectfully submitted,

John L. Rogitz
Registration No. 33,549
Attorney of Record
750 B Street, Suite 3120
San Diego, CA 92101
Telephone: (619) 338-8075

JLR:jg

APPENDIX A - APPEALED CLAIMS

6. A computer program device, comprising:

a computer program storage device including a program of instructions usable by an encryption computer, comprising:

logic means for chaining a data block to a plain text version of an adjacent block in the stream to render a chained block;

logic means for scrambling the chained block using a first round of a cipher to render a scrambled block; and

logic means for iterating the means for scrambling and chaining using subsequent rounds of the cipher.

7. The computer program device of Claim 6, wherein the means for iterating iterates forward and backward through the stream, using successive rounds of the cipher.

8. A computer system for encrypting a stream of data blocks, comprising a processor programmed to execute method acts including:

(a) receiving a sequence of N blocks;
(b) initializing a previous block variable B;
(c) for i=1 to N, executing a DO loop comprising:
(c)(1) XORing an ith block with B to render a modified ith block;
(c)(2) setting B equal to the modified ith block;
(c)(3) scrambling the modified ith block using at least one round of a cipher;
(c)(4) incrementing "i" by unity and returning to act (c)(1);
(d) initializing a previous block variable B;
(e) for i=N to 1, executing a DO loop comprising:
(e)(1) XORing an ith block with B, yielding a modified ith block;
(e)(2) setting B to the modified ith block;
(e)(3) scrambling the modified ith block using at least one next round of a cipher;
(e)(4) decrementing "i" by unity and returning to act (c)(1); and
(f) determining whether a predetermined number of iterations have been executed, and if not, returning to act (b) using a next round of the cipher, otherwise outputting an encrypted stream of data blocks.

9. The computer system of Claim 8, wherein the stream of data blocks is established by a computer program.

10. The computer system of Claim 9, wherein a respective round of the cipher is used for each iteration.

11. A method for generating a tamper resistant version of a software program including a stream of data blocks, comprising:

providing a cipher defining rounds;

iterating through the rounds of the cipher by iterating through respective outer loops of forward plain text chaining followed by backward plain text chaining; and

during each forward portion of an outer loop, applying a respective round of the cipher to each block, and during each backward portion of an outer loop, applying a respective round of the cipher to each block.

12. The method of Claim 11, further comprising:

(a) receiving a sequence of N blocks;

(b) initializing a previous block variable B;

(c) for i=1 to N, executing a DO loop comprising:

(c)(1) XORing an ith block with B to render a modified ith block;

(c)(2) setting B equal to the modified ith block;

(c)(3) scrambling the modified ith block using at least one round of a cipher;

(c)(4) incrementing "i" by unity and returning to act (c)(1);

(d) initializing a previous block variable B;

(e) for i=N to 1, executing a DO loop comprising:

(e)(1) XORing an ith block with B, yielding a modified ith block;

(e)(2) setting B to the modified ith block;

(e)(3) scrambling the modified ith block using at least one next round of a cipher;

(e)(4) decrementing "i" by unity and returning to act (c)(1); and

(f) determining whether a predetermined number of iterations have been executed, and if not, returning to act (b) using a next round of the cipher, otherwise outputting an encrypted stream of data blocks.

13. A method for generating a tamper resistant version of a software program including a stream of data blocks, comprising:

scrambling a block using one and only one round of a cipher; then

chaining the block to another block to render a chained block; then

scrambling the chained block using one and only round of the cipher.

14. The method of Claim 13, further comprising:

(a) receiving a sequence of N blocks;

(b) initializing a previous block variable B;

(c) for i=1 to N, executing a DO loop comprising:

(c)(1) XORing an ith block with B to render a modified ith block;

(c)(2) setting B equal to the modified ith block;

(c)(3) scrambling the modified ith block using at least one round of a cipher;

(c)(4) incrementing "i" by unity and returning to act (c)(1);

(d) initializing a previous block variable B;
(e) for i=N to 1, executing a DO loop comprising:
(e)(1) XORing an ith block with B, yielding a modified ith block;
(e)(2) setting B to the modified ith block;
(e)(3) scrambling the modified ith block using at least one next round of a cipher;
(e)(4) decrementing "i" by unity and returning to act (c)(1); and
(f) determining whether a predetermined number of iterations have been executed, and if not, returning to act (b) using a next round of the cipher, otherwise outputting an encrypted stream of data blocks.

15. A computer system for decrypting a stream of data blocks, comprising a processor programmed to execute method acts including:
(a) receiving a sequence of N blocks;
(b) for i= N to 1, executing a DO loop comprising:
(b)(1) reverse XORing an i^{th} block with a $block_{i-1}$;
(b)(2) unscrambling the i^{th} block using a round of a cipher to render an unscrambled block;
(b)(3) determining whether a $block_{i-1}$ exists, and if not, proceeding to act (c), otherwise;
(b)(4) decrementing "i" by unity and returning to act (b)(1);
(c) for i= 1 to N, executing a DO loop comprising:
(c)(1) reverse XORing an i^{th} block with a $block_{i+1}$;
(c)(2) unscrambling the i^{th} block using a single round of a cipher to render an unscrambled block;
(c)(3) determining whether a $block_{i+1}$ exists, and if not, proceeding to act (d), otherwise;
(c)(4) incrementing "i" by unity and returning to act (c)(1);
(d) determining whether a predetermined number of iterations have been executed, and if not, returning to act (b) using a next round of the cipher, otherwise outputting a decrypted stream of data blocks.

16. The computer system of Claim 15, wherein the stream of data blocks is established by a computer program.

17. The computer system of Claim 16, wherein a respective round of the cipher is used for each iteration.